Forward and Inverse Kinematics of 4-DoF SCARA: Using Optimization Algorithms

Mahdi Zavar[1], Niki Manouchehri[2],Alireza Safa[3*]

Abstract–In this article, solving and optimizing the problem of forward and inverse kinematics of SCARA is studied. This robot belongs to series robots and it has four degrees of freedom. First, we specify the coordinate axes for each joint and use it to extract the Danavit-Hartenberg parameters. Next, we examineforward kinematics of the robot and obtain the rotation matrices and the homogeneous transformation matrix and calculate the forward kinematics of the robot. Next, the method of solving the inverse kinematics problem of the robot is studied using different algorithms, including Cultural Algorithm, Genetic-Hybrid Algorithm, Gray Wolf Optimization, Firefly Algorithm, Ant Colony Optimization and Particle Swarm Optimization.Them, we optimize the inverse kinematics of the robot using these algorithms in two ways: fixed point and circular path. In the end, the effectiveness of the proposed approaches for solving the inverse kinematics problem of the SCARA robot is evaluated with multiple simulations.

Keywords:SCARA, Forward kinematics, Inverse kinematics, Optimization algorithms, Robot arms

1. Introduction

Robot arms are devices that consist of a number of joints and interfaces and have the ability to be used in a variety of ways, just like a human arm. Robotic arms in robotics industry and science are generally used in places where the human arm is either unable to do it or has limitations. Remote areas, the need for high accuracy and speed, reducing errors and possible risks are among the reasons for using these robotic arms. The use of such robots can be summarized according to their final implementation in the fields of moving parts, welding, cutting, etc.

SCARA is an industrial robot arm. The English word SCARA means selective adaptation assembly robot arm. This robot has four degrees of freedom, which means it has four joints. The first, second and fourth joints are rotary and the third joint is sliding. The advantages of SCARA robot include very fast movement, low space occupation, lower cost compared to four degrees of freedom robots, and high repeatability. Normally, this robot, like other robots, has disadvantages, including a higher price than Cartesian robots and the need for inverse kinematics controller software to interpolate linear movements [1].

Robot kinematics refers to the exploration of robots' motion concerning their internal joints, disregarding the impact of any external forces or torques acting on the robot system. It revolves around measuring the location, pace, and acceleration of different components of the robot in relevance to its coordinate systems. Robot kinematics encompasses two fundamental branches - forward and inverse kinematics. Forward kinematics determines the orientation and location of a robot's end-effector based on joint angles, while inverse kinematics denotes determining joint angle requirements to establish the desired location and direction of the robot's end-effector. This field plays an integral part in robot control, design, and programming. Robot kinematics enables engineers to program robotic movements and conducts simulation and testing to assure the accuracy of robotic movements. The robotic industry is experiencing significant progress in automation and autonomous robotic systems due to developments in robot kinematics. The study of robot kinematics empowers researchers and engineers to design and understand efficient, secure, and effective robotic systems in various fields of application [2,3].

Inverse kinematics is a crucial concept in robotics with a variety of applications. Its importance can be seen in four main areas: controlling robotic movements, planning a robot's path, avoiding collisions with the environment and other robots, and enhancing human-robot interaction. By understanding how the various actuators in a robot arm

[1]Department of Electrical Engineering, Faculty of Engineering, Golestan University, Gorgan, Iran.Email:m.zavar400@stu.gu.ac.ir

[2]Department of Electrical Engineering, Faculty of Engineering, Golestan University, Gorgan, Iran. Email:n.manouchehri400@stu.gu.ac.ir

[3*]**Corresponding Author:**Department of Electrical Engineering, Faculty of Engineering, Golestan University, Gorgan, Iran. Email:a.safa@gu.ac.ir

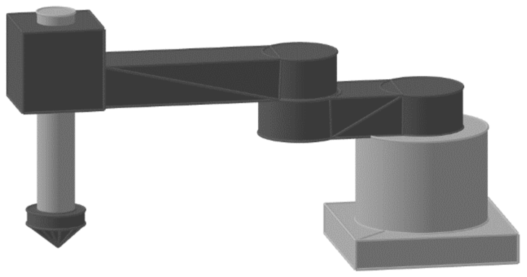
Received: 2023.08.03, Accepted: 2024.01.20

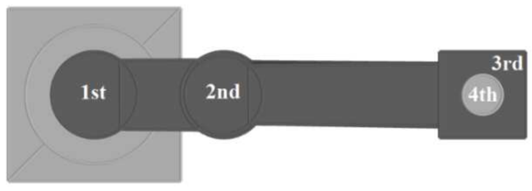
should move to achieve a specific end effector position, inverse kinematics ensures that robots can perform complex and precise movements accurately and safely. Overall, it is a vital component of robotics and automation. In the past, there were different mathematical methods used to solve inverse kinematics for robots. These methods include geometric-based [4-6], numerical-based [7-9], artificial intelligence-based [10-13] and dynamic-based approaches [14-16]. These methods have evolved and improved over time with the advancement of technology and enhancement in computational power.

In this article, the robot and its parts are first introduced, then one of the most famous laws, known as Denavit-Hartenberg laws, is stated for calculating and extracting the forward kinematics of the robot. In the following, using the defined rules, the coordinate axes are assigned to the robot's joints and a table of the robot's parameters is prepared. Using the existing relationships, the coordinates of the final robot operator will be obtained parametric. In the next section, the inverse kinematics of the robot is discussed. In this section, a brief explanation of the algorithms is given, and by using the robot workspace and defining a cost function, the necessary simulations are performed to compare the mentioned algorithms. Finally, the necessary evaluations are done to find the most optimal algorithm.

2. Robot Parts

The joints of this robot can be moved and rotated by AC servo motors, in which a gearbox is used in the rotational joints and a ball screw is used in the transfer joint. In the selected example, the end effector of the robot is an electromagnetic clamp that has the ability to grab and drop objects for assembly work. The robot and its joints are shown in Fig. 1.


(a)



(b)

Fig.1.SCARA; (a) 3D schematic, (b) Joints

3. Denavit-Hartenberg Convention

According to the Denavit-Hartenberg (D-H) laws, the coordinate axes assigned to a robot's joints must prioritize two rules. First, the new x-axis should be perpendicular to both the previous z-axis and the new z-axis. Second, the coordinate axes for a robot's joints should abide by these rules. A matrix is determined for each joint of the robot which expresses the joint's position relative to the initial frame. These parameters are then used to compute a transformation matrix between nearby coordinate frames, which determines the position and orientation of the robot's end-effector according to its base [17,18]. The forward kinematics of the robot can be derived by using D-H parameters and transformation matrices, which calculate the configuration of the robot's end-effector given its joint angles [19,20]. This matrix can be obtained by multiplying four rotation and transfer matrices together follows:

$$T_i = Rot_{z,\theta_i}\, Trans_{z,d_i}\, Trans_{x,a_i}\, Rot_{x,\alpha_i} \tag{1}$$

the expansion of (1) is written as follows:

$$T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\cos\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

4. Extraction of Forward Kinematics

In forward kinematics, the parameters a_i, d_i, α_i and

θ_i result in the end-effector's final position [21]. The framework of the axes related to the joints and the end-effector is calculated by D-H laws [22].

According to D-H laws, as stated, coordinate axes are considered for the robot in Fig. 2, and these axes will be used for calculations.

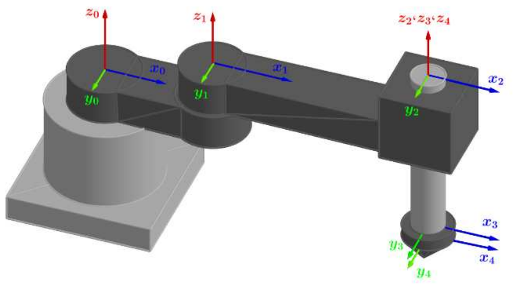


Fig. 2. Assign coordinate frame to robot joints

After selecting the coordinate axes for the robot joints, the D-H table of the robot is formed.

Table 1. D-H table and SCARA's parameters

Link	$a_i\ (m)$	$\alpha_i(°)$	$d_i\ (m)$	$\theta_i(°)$
1	$a_1 = 0.2$	0	0	$\theta_1: (\pm 120)$
2	$a_2 = 0.2$	0	0	$\theta_2: (\pm 120)$
3	0	0	$-d_3: (0, .048)$	0
4	0	0	$-d_4 = 0.05$	$\theta_4: (\pm 120)$

Using D-H parameters, the homogeneous transformation of the robot can be calculated as follows [23]:

$$T_4^0 = \begin{bmatrix} \cos\theta_{124} & -\sin\theta_{124} & 0 & a_1\cos\theta_1 + a_2\cos\theta_{12} \\ \sin\theta_{124} & \cos\theta_{124} & 0 & a_1\sin\theta_1 + a_2\sin\theta_{12} \\ 0 & 0 & 1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$\begin{aligned} \theta_{12} &= \theta_1 + \theta_2 \\ \theta_{124} &= \theta_1 + \theta_2 + \theta_4 \end{aligned} \quad (4)$$

Therefore, according to the definitions made for D-H rules and robot parameters, the coordinates of the end-effector can be extracted from the above matrix.

$$\begin{aligned} x_{ee} &= a_1\cos\theta_1 + a_2\cos\theta_{12} \\ y_{ee} &= a_1\sin\theta_1 + a_2\sin\theta_{12} \\ z_{ee} &= -d_3 - d_4 \end{aligned} \quad (5)$$

5. Inverse Kinematics

To determine the joint parameters of the robot, the final coordinates of the robot's end-effector, which are found in the third column of (3), must be known. Once the final position is determined, optimization algorithms can be used to determine the most optimal way to reach the desired position based on the various possible ways [24].

The primary objective of the general constraint optimization problem is to determine the value of x such that:

$$minimize\ f(x), \quad x = (x_1, \dots, x_m) \quad (6)$$

subject to

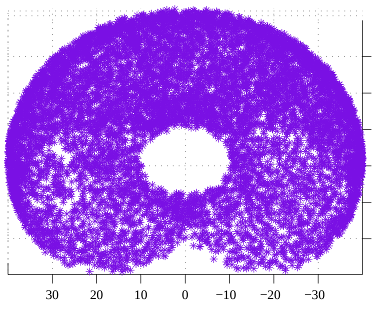
$$\begin{aligned} h(x) &= 0 \\ g(x) &\le 0 \end{aligned} \quad (7)$$

where the problem's constraints, $h(x)$ and $g(x)$, may be linear or nonlinear. The set $h(x)$ represents p equality constraints, whereas $g(x)$ signifies q inequality constraints.

To obtain the most optimal way of reaching a desired position, optimization algorithms can be employed and the cost function described in (2) can be utilized. This cost function calculates the difference between the current coordinates and the final coordinates, and the aim is to minimize the difference between these two values in order to find the optimal solution [25]. As an example, the final coordinates (-0.23, 0.20, -0.20) have been selected from the robot workspace shown in Fig. 3. In this article, different optimization algorithms are used to calculate the inverse kinematics of the robot.

$$\text{cost} = \sqrt{(x_{out} - x_{ee})^2 + (y_{out} - y_{ee})^2 + (z_{out} - z_{ee})^2} \quad (8)$$

where, x_{ee}, y_{ee} and z_{ee} are coordinates of the end-effector and $(x_{out}, y_{out}, z_{out})$ is the algorithms answer.


(a)

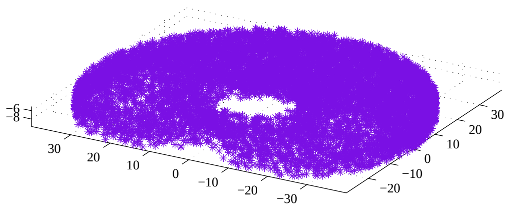

(b)

Fig. 3. SCARA workspace; (a) $x-y$ display, (b) 3D display

Cultural Algorithms (CA) are based on the fundamental concept of obtaining problem-solving knowledge, or beliefs, from the developing population, and employing that knowledge to direct the search process (Algorithm1). CA is equivalent to GA, with the difference that instead of biological evolution, cultural evolution is used to optimize the model. Another difference between these two algorithms is the time of evolution, biological evolution happens over many years and cultural evolution happens in a much shorter time [26].

Algorithm 1. Cultural Algorithm

```
Require: Data, Parameters
1. Begin
2. $n \leftarrow 0$
3. $d \leftarrow \text{GetData}(Data)$
4. $Chromosome \leftarrow \text{ConvertData}(d)$
5. $Belief_n \leftarrow \text{InitializaBeliefSpace}(d, Chromosome)$
6. $Pop_n \leftarrow \text{PopulationSpace}(n, Chromosome)$
7. $Eva_{Pop} \leftarrow \text{EvaluationPop}(n, Pop_n)$
8. while stop condition do
9.     $n \leftarrow n+1$
10.    foreach agent do
11.        Reproduction by social trait
12.        UpdatePop$(n, Pop_n)$
13.    end for
14.    $Eva_{Pop} \leftarrow \text{EvaluationPop}(n, Pop_n)$
15.    $Pop_n \leftarrow Pop_{n-1}$
16. Accept $Pop_n$
17. UpdateBeliefSpace$(n, Belief_n)$
18. end while
19. return $Pop_n, Belief_n$
```

As its name suggests, the Gray Wolf Algorithm (GWO) is inspired by nature and performs optimization based on the hierarchical structure of gray wolves during hunting (Algorithm2). The algorithm starts by randomly initializing a population of solutions, with each solution representing a wolf. GWO consists of four major steps, which include hunting for prey, following the alpha wolf, following the beta wolf, and following the delta wolf [27, 28].

Algorithm 2. Gray Wolf Optimization

```
Require: $Iteration, Dim, \alpha, \beta, \gamma$
1. $D \leftarrow C.X_p(t) - X(t)$
2. $X(t+1) \leftarrow X_p - A.D$
3. $A \leftarrow 2a.r - a, \quad a: [2 \rightarrow 0]$
4. $C \leftarrow 2.r, \quad r: [0,1]$
5. $Pop \leftarrow \text{InitializeGWOPop}()$
6. $Fitness \leftarrow \text{FitnessFunction}(x)$
7. $\alpha := FirstBest_{sol}$
8. $\beta := SecondBest_{sol}$
9. $\gamma := ThirdBest_{sol}$
10. while $t \leq Iteration$ do
11.     foreach agent do
12.         Update positions: $\alpha, \beta, \gamma$
13.     end for
14.     Update $a, A, C$
15.     Find fitness of all wolves
16. Update $\alpha, \beta, \gamma$
17.     $t \leftarrow t+1$
18. end while
19. return $\alpha$
```

In contrast to genetic algorithms, evolutionary programming, and evolution strategies, Particle Swarm Optimization (PSO) does not utilize a selection operation (Algorithm3). Throughout the duration of the run, which is defined as the total number of evolutionary algorithm generations prior to termination, all particles in PSO remain as members of the population. The updated velocity of the particle is determined by considering both its own previous best position and the previous best position of its companions. Afterwards, the particle flies using its updated velocity. One noteworthy aspect of PSO is that it does not rely on the survival of the fittest mechanism, which sets it apart from other evolutionary algorithms [29].

Algorithm 3. Particle Swarm Optimization

```
Require:$Iteration, Dim, NOP$
1. $P_{s1} \leftarrow a_1\ (ownway)$
2. $P_{s2} \leftarrow a_2\ (P_{Best})$
3. $P_{s3} \leftarrow a_3\ (G_{Best})$
4. $Pop \leftarrow$ InitializePSOPop()
5. while $t \leq Iteration$ do
6. foreach particle $P$do
7.         $Value_p \leftarrow$ Evaluation($x_p$)
8. if $Value(x_p) \leq Value(P_{Best})$then
                   9. $P_{Best} \leftarrow x_p$
10. end if
11.     if $Value(x_p) \leq Value(G_{Best})$then
12.         $G_{Best} \leftarrow x_p$
13.     end if
14.     end for
15. foreach particle $P$ do
16.         $Velocity_p \leftarrow$ DefineVelocity($P_{s1}, P_{s2}, P_{s3}$)
17.$x_p \leftarrow$ UpdateParticle($x_p, Velocity_p$)
18. end for
19.   Update probabilities
20. end while
21. return$P_{Best}$
```

The Genetic-Hybrid Algorithm (G-HA) is an evolutionary algorithm that is formulated based on biological methods including(Algorithm4): mutation, inheritance, selection principles, etc. and is used for prediction and mathematical modeling, to which the term hybrid has been added in this article to give a better answer. G-HA, designed to replicate specific natural evolutionary processes, has become a highly effective stochastic search technique that relies on the principles of natural selection and genetics. G-HA commences with a set of arbitrary solutions called the population, where each individual is coded as a chromosome that proposes a solution for the problem. These chromosomes evolve through multiple iterations or generations. During each generation, the chromosomes undergo evaluation based on certain fitness measures. As the process progresses through several generations, the algorithm converges to the best chromosome, signifying the optimal solution. Once the cost reaches to 10^{-4}, the Hybrid function is employed to obtain final solution[30].

Algorithm 4. Genetic-Hybrid Algorithm

```
1. float, int, bin, etc. ← coding
2. $Cost \leftarrow$ CostFunction ()
3. $Pop \leftarrow$ GeneratePopulation()
4. whileterminating conditiondo
5.      Selection operator → soft or hard mode
6. if$n_{pop}$not generatethen
7.        Roulette wheel, Rank, Top, etc.
8. New generation
                 9. Apply crossovr
10. Apply mutation
11.     end if
12. Apply Hybric Function
13.     Find $Best_{cost}$
14. end while
15. return $Best_{cost}$
```

The Firefly Algorithm (FA) is a meta-heuristic algorithm that originates from the communication between fireflies (Algorithm5). The cooperation and competition of each less intelligent member of the population with another creates a higher order of intelligence. FA focuses on generating new solutions within a search space through its heuristic, known as the 'lower level', and selects the optimal solution for survival. To prevent the solution from being trapped in local optima, the algorithm utilizes randomization in its arch process. Additionally, the local search function continually improves a candidate solution until it reaches a local optimum [31].

Algorithm 5. Firefly Algorithm

```
Require:$Iteration, Dim$
1. $Pop \leftarrow$ InitializeFAPop()
2. $Cost \leftarrow$ CostFunction($x$)
3. $I \leftarrow$ IntensityPop($Cost$)
4. while $t \leq Iteration$ do
5. for $i = 1{:}n$do
                  6. for $j = 1{:}n$do
7. if$I_j \leq I_i$then
            8. Firefly($j, Dim$) ← Firefly($i, Dim$)
9. end if
10.    Evaluation new solution
11.         Update intensity
12. end for
13. end for
14.  Find the current $Best_{sol}$
15. end while
16. return $Best_{sol}$
```

According to the behavior of ants in finding the closest path to food, the Ant Colony Optimization (ACO) has been formed (Algorithm6). Ants randomly follow paths to find food and leave a substance called Pheromone so that the next ants reach the food, but on the way back to the nest, they may follow a non-directive path, and the higher the amount of Pheromone accumulated, ants are more attracted to it. In the same way, ACO employs a comparable process to solve optimization problems [32].

Algorithm 6. Ant Colony Optimization

```
**Require:**$It, Dim, \rho, \alpha, \beta, NOP$
1. $Pop_{Best} \leftarrow$ CreatSolution($Dim$)
2. $Cost_{PopBest} \leftarrow$ Cost($S_h$)
3. $Ph \leftarrow$ InitializationPh($Cost_{PopBest}$)
4. **while**Stop condition**do**
5. **for** $i = 1$: It**do**
6. $S_i \leftarrow$ StepwiseMovement($\alpha, \beta, Ph, Dim$)
7.         $Cost_{Si} \leftarrow$ Cost($S_i$)
               8. **if** $Cos_{PopBest} \geq Cost_{Si}$**then**
9.             $Cos_{PopBest} \leftarrow Cost_{Si}$
10.      $Pop_{Best} \leftarrow S_i$
11.        **end if**
12. **end for**
13. OldPh($Ph, \rho$)
14. UpdatePh($Ph, S_i, Cost_{Si}$)
15. **end while**
16. **return** $Pop_{Best}, Cost_{PopBest}$
```

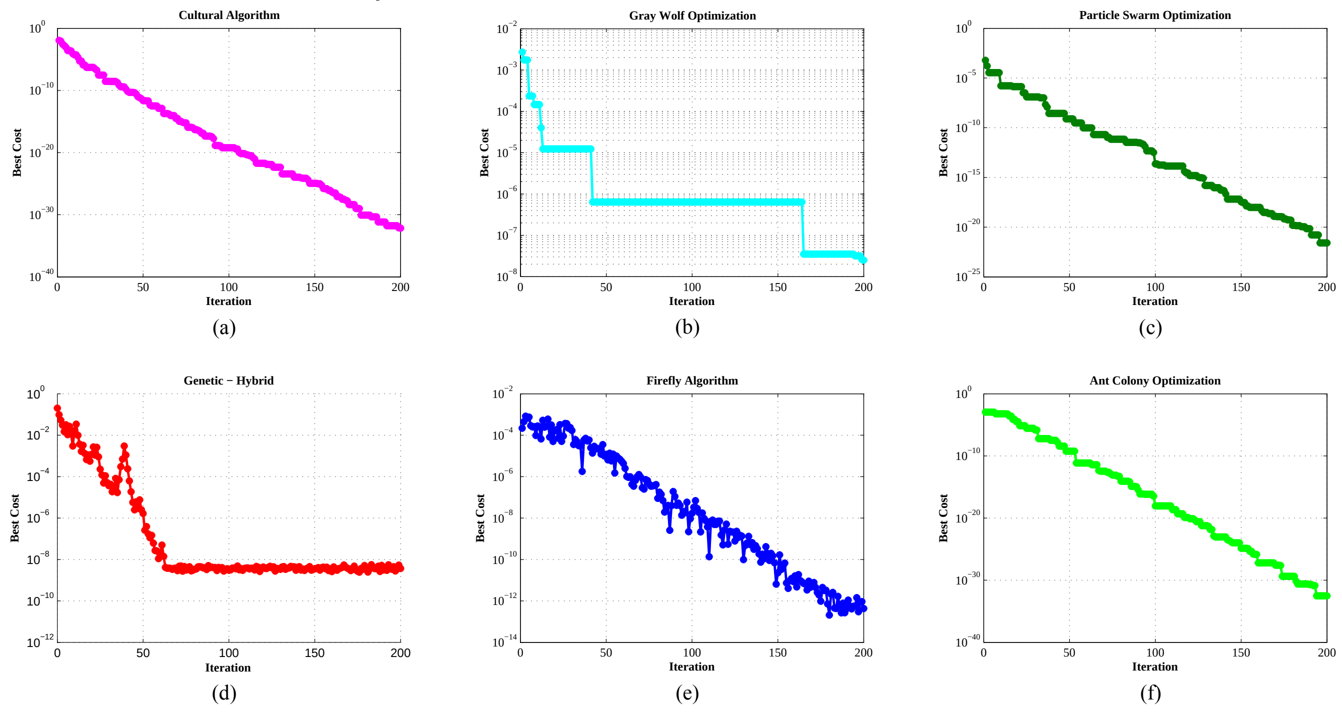


Fig.4. The cost function for 200 iterations; (a) CA , (b) GWO, (c) PSO, (d) G-HA, (e) FA, (f) ACO

6. Discussion

This section includes tables and graphs generated from MATLAB software [1] during the calculation of inverse kinematics. Fig. 4 displays graphs of the variant cost function after 200 iterations. Each of the algorithms includes different parameters, such as the distribution of search agents, population count, and the percentage of different tasks, among others. Two parameters that have a significant impact on the execution time are the lower band and the upper band. These parameters are used to limit the search environment.

Figure 4 displays the cost functions for various optimization algorithms. In these graphs, the cost function for CA shown in Fig. 4(a) displays a continuous downward path without any stops, indicating a consistent decrease in the value of the cost function. In G-HA, as shown in Fig. 4(d), a broad solution space is first evaluated. Once theerror reaches the 10^{-3} threshold, the Nelder-Mead Simplex algorithm is employed to obtain the final solution. Similarly, the cost function for FA shown in Fig. 4(e) also exhibited

[1]System info: Intel(R) Core(TM)i3 CPU, 2.4GHz, 4GB RAM

many changes but maintained a general downward trend. On the other hand, GWO shown in Fig. 4(b) experienced horizontal stops in some iteration, indicating that the cost function remained constant during these iterations. In the case of PSO and ACO shown in Fig. 4(c) and Fig. 4(f), respectively, the cost function trended downwards in a continual fashion.

To facilitate comparison of the different algorithms used, Table 2 displays the recorded coordinate error, cost function values, and execution times for different population sizes. As expected, the execution time of the algorithms increases with an increase in population size. The coordinate error values in the table are expressed in meters, while time is measured in seconds.

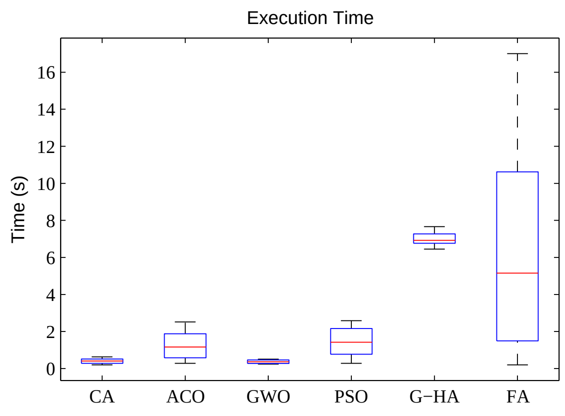


Fig.5. Graph of execution time in mentioned algorithms from Table

The inverse kinematics for a single point in the SCARA workspace was optimized using different populations of the specified algorithms. Now, we will assess each algorithm's response to the robot operator's movement along a circular path in the workspace, as shown in Fig. 6. This route covers 11 points, each with its own specific coordinates, listed in Table 3. To ensure a fair comparison between them, each point was optimized by the algorithms under the same conditions.

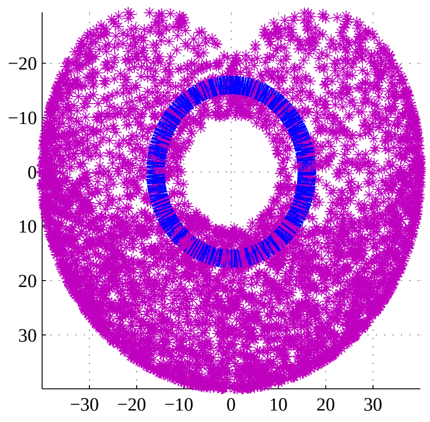
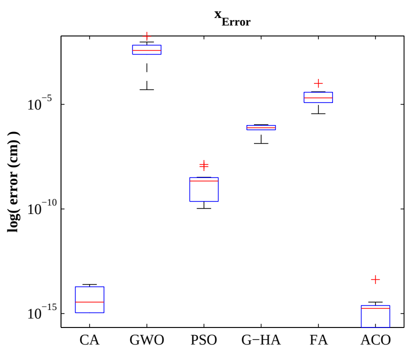
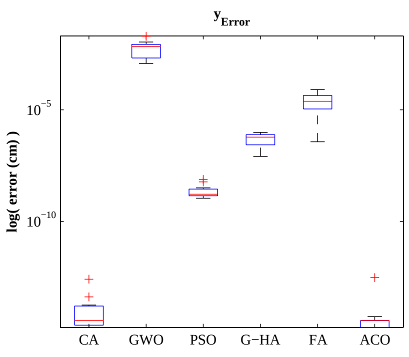


Fig. 6. Circular path in the SCARA workspace

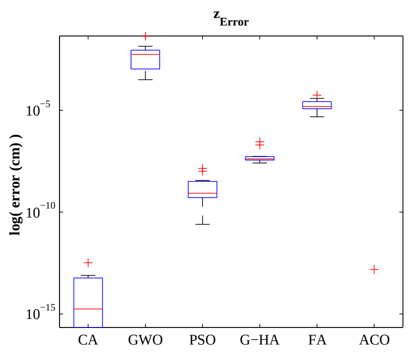
The charts representing absolute value of$x$, y, and z errors, as well as the cost function, utilize a logarithmic vertical axis due to the small values of the answers and the



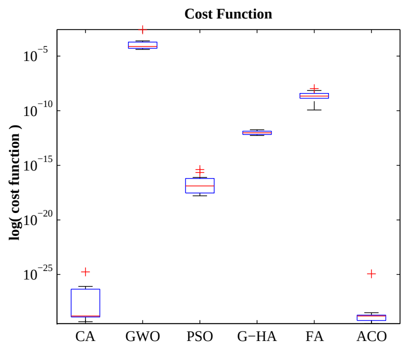
(a)



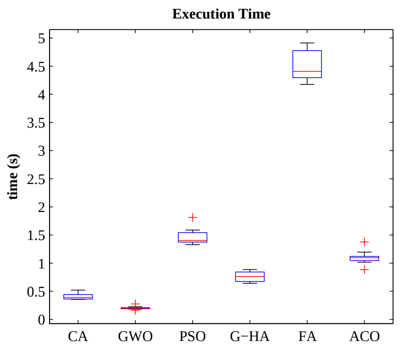
(b)



(c)



(d)



(e)

Fig. 7. Algorithms results for circular path; (a) Absolute value of x-error, (b) Absolute value of y-error, (c) Absolute value of z-error, (d) Cost function, (e) Execution time

significant differences in the errors. This choice allows for better visualization and diagnosis. Each graph depicts the highest value with a blue line at the top of the rectangle, the lowest value with a blue line at the bottom of the rectangle, and the average value with a red line in the middle of the rectangle.

The first analysis pertains to the absolute value of x error, as shown in Fig. 7(a), which presents the algorithms' errors for the robot's selected path points listed in Table 3. This graph reveals that the average response of the algorithms from the lowest error to the highest error is: ACO, CA, PSO, G-HA, FA, and GWO.

Similarly, Fig. 7(b) and Fig. 7(c) demonstrate the absolute value of y and z errors, respectively, with the only difference being the change in the position of CA and ant ACO. In Fig. 7 (d), according to the cost function expressed in (8), the average response of the CA and ACO is almost identical.

Finally, Fig. 7(e) illustrates the execution time of each algorithm for the 11 selected points, presented in the following order of algorithm time: GWO, CA, G-HA, PSO, ACO, and FA.

Table 3. Points in circular path

Axis	x	y	z
1	-14	7.75	-6
2	-8	13.85	-6
3	0	16	-6
4	8	13.85	-6
5	14	7.75	-6
6	16	0	-6
7	14	$7.75-$	-6
8	8	-13.85	-6
9	0	-16	-6
10	-8	-13.85	-6

Table 2. Comparison of algorithms

Agents	Algorithms	$\|Error_x\|(m)$	$\|Error_y\|(m)$	$\|Error_z\|(m)$	Cost	Time (s)
20	CA	6.0518×10^{-3}	2.2313×10^{-3}	8.6302×10^{-4}	4.2349×10^{-5}	0.24493
	GWO	5.1071×10^{-4}	1.4031×10^{-4}	7.6245×10^{-4}	8.6185×10^{-7}	0.27279
	PSO	1.8112×10^{-10}	2.7348×10^{-10}	8.1221×10^{-11}	1.1419×10^{-19}	0.53402
	G-HA	8.3137×10^{-8}	5.3925×10^{-8}	1.4150×10^{-8}	2.1700×10^{-13}	6.82576
	FA	6.7710×10^{-6}	4.0840×10^{-6}	1.0400×10^{-6}	2.9151×10^{-12}	0.69410
	ACO	2.8479×10^{-13}	3.4900×10^{-10}	5.7465×10^{-10}	4.5203×10^{-19}	0.42001
40	CA	2.775×10^{-17}	0	0	7.7037×10^{-34}	0.28883
	GWO	3.9048×10^{-4}	7.3600×10^{-4}	1.1550×10^{-3}	2.0282×10^{-6}	0.34990
	PSO	1.8920×10^{-10}	7.8578×10^{-11}	2.6544×10^{-12}	4.1978×10^{-20}	1.06167
	G-HA	1.0807×10^{-7}	1.6725×10^{-8}	6.3240×10^{-9}	1.2600×10^{-13}	6.86675
	FA	5.3260×10^{-6}	3.4680×10^{-6}	3.2500×10^{-7}	5.5948×10^{-13}	2.95924
	ACO	3.7062×10^{-12}	3.80451×10^{-12}	6.5702×10^{-13}	2.8642×10^{-23}	0.74903
60	CA	0	2.7755×10^{-17}	5.5511×10^{-17}	3.8518×10^{-33}	0.46891
	GWO	3.4591×10^{-4}	2.3062×10^{-4}	6.0688×10^{-3}	3.7003×10^{-5}	0.38846
	PSO	6.0043×10^{-12}	4.8507×10^{-11}	1.1871×10^{-10}	1.6481×10^{-20}	1.54468
	G-HA	2.1609×10^{-7}	3.7832×10^{-8}	1.3569×10^{-7}	3.8800×10^{-13}	7.26907
	FA	8.1540×10^{-7}	3.7400×10^{-7}	4.0580×10^{-7}	9.6949×10^{-13}	6.12580
	ACO	2.3869×10^{-15}	6.7890×10^{-14}	9.0843×10^{-14}	1.2867×10^{-26}	1.28537
80	CA	0	2.7755×10^{-17}	0	7.7037×10^{-34}	0.51326
	GWO	1.2172×10^{-4}	1.4443×10^{-4}	1.6078×10^{-3}	2.6208×10^{-6}	0.47053
	PSO	2.6821×10^{-12}	6.4868×10^{-12}	1.7253×10^{-11}	3.4694×10^{-22}	2.15798
	G-HA	6.8448×10^{-8}	2.3166×10^{-7}	2.1532×10^{-7}	1.5900×10^{-13}	7.04521
	FA	6.9800×10^{-8}	2.8630×10^{-7}	6.2980×10^{-7}	4.8352×10^{-13}	10.62385
	ACO	1.8318×10^{-15}	7.5495×10^{-15}	3.3306×10^{-16}	6.0461×10^{-29}	1.87799
100	CA	2.775×10^{-17}	5.5511×10^{-17}	3.6082×10^{-16}	1.3404×10^{-31}	0.63141
	GWO	1.4823×10^{-4}	1.4343×10^{-4}	3.1004×10^{-3}	9.6551×10^{-6}	0.50475
	PSO	8.6114×10^{-12}	1.4992×10^{-11}	8.4837×10^{-13}	2.9964×10^{-22}	2.58533
	G-HA	4.5212×10^{-9}	8.5804×10^{-9}	3.6674×10^{-8}	2.7000×10^{-13}	7.66663
	FA	7.5200×10^{-8}	2.4940×10^{-7}	4.3770×10^{-7}	2.5940×10^{-13}	17.0053
	ACO	0	0	5.5511×10^{-17}	3.0814×10^{-33}	2.52019

7. Conclusion

Based on Fig. 5, which is drawn using the final column of Table 2 and considers a fixed point with the change of population as the working criterion, it is apparent that the FA and G-HA algorithms lack time optimization attributes, and thus they were excluded from comparison. Moreover, the GWO and CA algorithms are top-performing options in terms of execution time optimization. However, when it comes to coordinate error, the CA, PSO, and ACO algorithms are preferable. Although the value of the cost function is satisfactory in all of the algorithms except for GWO.

Evaluating these three performance criteria together, we can conclude that the CA algorithm is the best among the existing algorithms for the SCARA's inverse kinematics optimization. Using a fixed population, as listed in Table 3, CA algorithm outperformed ACO in terms of time chart while both algorithms achieved the same amount of coordinate error. Therefore, CA demonstrated relative superiority over other algorithms in both analyses and yielded the best results for optimizing robot kinematics.

References

[1] Roshanianfard A., Mengmeng D. and Nematzadeh S., "A 4-dof scara robotic arm for various farm applications: Designing, kinematic modelling, and parameterization."*ActaTechnologicaAgriculturae*, Vol. 24. No. 2, pp. 61–66, 2021.

[2]Liu Y., Wan M., Xing W.J., Xiao Q.B. and Zhang W.H., "Generalized actual inverse kinematic model for compensating geometric errors in five-axis machine tools." *International Journal of Mechanical Sciences*, Vol. 145, pp. 299–317, 2018.

[3]My C.A. and Bohez E.J., "A novel differential kinematics model to compare the kinematic performances of 5-axis cnc machines." *International Journal of Mechanical Sciences*, Vol. 163, pp. 105117, 2019.

[4] Xiao W., Liu C., Hu D., Yang G. and Han X., "Soft robotic surface enhances the grasping adaptability and reliability of pneumatic grippers."*International Journal of Mechanical Sciences*, Vol. 219, pp. 107094, 2022.

[5] Mu˜noz J., L´opez B., Quevedo F., Barber R., Garrido S. and Moreno L., "Geometrically constrained path planning for robotic grasping with differential evolution and fast marching square."*Robotica*, Vol. 41, No. 2, pp. 414–432, 2023.

[6] Yang Ch., Geng Sh., Walker I., Branson D.T, Liu J., Dai J.S. and Kang R., "Geometric constraint-based modeling and analysis of a novel continuum robot with shape memory alloy initiated variable stiffness."The *International Journal of Robotics Research*, Vol. 39, No. 14, pp. 1620–1634, 2020.

[7] Wei Ch., Taghavifar H. and Mardani A., "Appraisal of numerical based finite element method to synthesise the wheel-obstacle collision dynamics using a single-wheel tester."*International Journal of Heavy Vehicle Systems*, Vol. 26, No. 3-4, pp. 578–598, 2019.

[8] Fang G., Tian Y., Yang Z.X., Geraedts J.M. and Wang C.C., "Efficient jacobianbased inverse kinematics with sim-to-real transfer of soft robots by learning."*IEEE/ASME Transactions on Mechatronics*, Vol. 27, No. 6, pp. 5296–5306, 2022.

[9] Hendriko H., Nurkhamdi J.J. and Imam M.M., "Analytical based inverse kinematics method for 5-axis delta robot."*International Journal of Materials, Mechanics and Manufacturing*, Vol. 6, No. 4, 2018.

[10] Chawla I., Pathak P., Notash L., Samantaray A., Li Q. and Sharma U.,"Inverse and forward kineto-static solution of a large-scale cabledriven parallel robot using neural networks."*Mechanism and Machine Theory*, Vol. 179, pp. 105107, 2023.

[11] Alsamhi S.H., Ma O. and Ansari M.S., "Survey on artificial intelligence based techniques for emerging robotic communication."*Telecommunication Systems*, Vol. 72, pp. 483–503, 2019.

[12] Ghith E.S. and Tolba F.A.A., "Labview implementation of tuning pid controller using advanced control optimization techniques for microrobotics system."*International Journal of Mechanical Engineering and Robotics Research*, Vol. 11, No. 9, 2022.

[13] Wang Ch., TeoTh.S. and Janssen M., "Public and private value creation using artificial intelligence: An empirical study of ai voice robot users in chinese public sector."*International Journal of Information Management*, Vol. 61, pp. 102401, 2021.

[14] Bai G., Liu L., Meng Y., Luo W., Gu Q. and Wang J., "Path tracking of wheeled mobile robots based on dynamic prediction model."*IEEE Access*, Vol. 7, pp. 39690–39701, 2019.

[15] Torres-Figueroa J., Portilla-Flores E.A., V´asquez-Santacruz J.A., Vega-Alvarado E. and Mar´ın-Ur´ıas L.F., "A novel general inverse kinematics optimization-based solution for legged robots in dynamic walking by a heuristic approach."*IEEE Access*, Vol. 11, pp. 2886–2906, 2023.

[16] Ghasemi A., Li P. and Xie W.F., "Adaptive switch image-based visual servoing for industrial

robots."*International Journal of Control, Automation and Systems*, Vol. 18, pp. 1324–1334, 2020.

[17] Susanto C., Limanuel F. and Rippun F., "Design and implementation of pose recording system with denavithartenberg method in a 6-dof robot rotaric."*WidyaTeknik*, Vol. 22, No. 1, pp. 1–8, 2023.

[18] Bahani A., Ech-Chhibat M.E., Samri H. and Elattar H.A., "The inverse kinematics evaluation of 6-dof robots in cooperative tasks using virtual modeling design and artificial intelligence tools."*International Journal of Mechanical Engineering and Robotics Research*, Vol. 12, No. 2, 2023.

[19] Khanesar M.A., Yan M., Isa M., Piano S. and Branson D.T., "Precision denavit–hartenberg parameter calibration for industrial robots using a laser tracker system and intelligent optimization approaches."*Sensors*, Vol. 23, No. 12, pp. 5368, 2023.

[20] Xu X., Bai Y., Zhao M., Yang J., Pang F., Ran Y., Tan Zh. and Luo M., "A novel calibration method for robot kinematic parameters based on improved manta ray foraging optimization algorithm."*IEEE Transactions on Instrumentation and Measurement*, Vol. 72, pp. 1–11, 2023.

[21] Zhen Sh.Ch., Ma M.C., Liu X.L., Chen F., Zhao H. and Chen Y.H., "Model-based robust control design and experimental validation of scara robot system with uncertainty."*Journal of Vibration and Control*, Vol. 29, No.1-2, pp. 91–104, 2023.

[22] Jeddi M., Khoogar A. and MehdipoorOmrani A., "Eye in-hand stereo image based visual servoing for robotic assembly and set-point calibration used on 4 dofscara robot."*International Journal of Robotics, Theory and Applications*, Vol. 8, No. 1, pp. 33–44, 2022.

[23] Rigatos G., Abbaszadeh M., Busawon K. and Pomares J., "Nonlinear optimal control for a 4-dof scara robotic manipulator."*Robotica*, pp. 1–54, 2023.

[24] Zhong G., Peng B. and Dou W., "Kinematics analysis and trajectory planning of a continuum manipulator."*International Journal of Mechanical Sciences*, Vol. 222, pp.107206, 2022.

[25] Xie Sh., Sun L., Wang Zh. and Chen G., "A speedup method for solving the inverse kinematics problem of robotic manipulators."*International Journal of Advanced Robotic Systems*, Vol. 19, No. 3, pp. 17298806221104602, 2022.

[26] Ali H.I., Hasan A.F. and Jassim H.M., "Optimal h2pid controller design for human swing leg system using cultural algorithm."*Journal of Engineering Science and Technology*, Vol. 15, No. 4, pp. 2270–2288, 2020.

[27] Shrivastava A. and Dalla V.K., "Jerk optimized motion planning of redundant space robot based on grey-wolf optimization approach."*Arabian Journal for Science and Engineering*, Vol. 48, No. 3, pp. 2687– 2699, 2023.

[28] Al-Tashi Q., Rais H.M, Abdulkadir S.J, Mirjalili S.A. and Alhussian H., "A review of grey wolf optimizer-based feature selection methods for classification."*Evolutionary Machine Learning Techniques: Algorithms and Applications*, pp. 273– 286, 2020.

[29] Sai H., XuZh., Xu C., Wang X., Wang K. and Zhu L., "Adaptive local approximation neural network control based on extraordinariness particle swarm optimization for robotic manipulators."*Journal of Mechanical Science and Technology*, Vol. 36, No. 3, pp. 1469–1483, 2022.

[30] Wang W., Tian G., Zhang H., Li Zh. and Zhang L., "A hybrid genetic algorithm with multiple decoding methods for energy-aware remanufacturing system scheduling problem."*Robotics and Computer-Integrated Manufacturing*, Vol. 81, pp. 102509, 2023.

[31] Kundra H., Khan W., Malik M., Rane K.P, Neware R. and Jain V., "Quantum-inspired firefly algorithm integrated with cuckoo search for optimal path planning."*International Journal of Modern Physics C*, Vol. 33, No. 02, pp. 2250018, 2022.

[32] Zhang T., Cheng Y., Wu H., Song Y., Yan Sh., Handroos H., Zheng L., Ji H. and Pan H., "Dynamic accuracy ant colony optimization of inverse kinematic (daacoik) analysis of multi-purpose deployer (mpd) for cfetr remote handling."*Fusion Engineering and Design*, Vol. 156, pp. 111522, 2020.